Math.Comput.Sci. (2016) 10:409–427
DOI 10.1007/s11786-016-0274-0

A Survey on Retrieval of Mathematical Knowledge

Ferruccio Guidi · Claudio Sacerdoti Coen

Received: 6 January 2016 / Revised: 16 March 2016 / Accepted: 20 June 2016 / Published online: 18 July 2016

Abstract We present a survey of the literature on indexing and retrieval of mathematical knowledge, with pointers to 77 papers and tentative taxonomies of both retrieval problems and recurring techniques.

1 Introduction

Retrieval of mathematical knowledge is always presented as the low hanging fruit of Mathematical Knowledge Management, and it has been addressed in several papers by people coming either from the formal methods or from the information retrieval community. The problem being resistant to classical content search techniques [35], it is usually addressed combining a small set of new ideas and techniques that are recurrent in the literature. Despite the amount of work, however, there is not a single solution that is clearly winning on the others, nor convincing unbiased benchmarks to compare solutions. Some authors like [32] also suggest that the community should first better understand the actual needs of mathematicians from an unbiased perspective to improve the mathematical knowledge management (MKM) technology as a whole. In this paper we collect a hopefully comprehensive bibliography, and we roughly classify the papers according to novel taxonomies both for the problems and the techniques employed. The only other surveys on the same topic are [22], which is a short version of this article [10], now outdated and focused mostly on (European) research projects that contributed to the topic in the 6th Framework Programme, [73], which covers less literature in much greater detail without attempting a classification [41], which is focused on evaluation of mathematics retrieval, and [40], which is written in Slovak.

We begin our discussion in Sect. 2 with a purpose driven taxonomy made of three different retrieval problems that deal with mathematical knowledge. Each problem is characterised by its own set of expectations and constraints, and adapting a solution to another problem may be infeasible or may yield poor results. Then we classify the papers according to an encoding based taxonomy (presentation vs content vs semantics), presented in Sect. 3, and to a taxonomy of techniques employed, presented in Sect. 4. In Sect. 5 we point to the rich literature relative to the problem of ranking, and in Sect. 6 we summarise the previous chapters presenting a reference architecture for a mathematical document retrieval system.

F. Guidi · C. Sacerdoti Coen (✉)
Department of Computer Science and Engineering-DISI, University of Bologna, Bologna, Italy
e-mail: claudio.sacerdoticoen@unibo.it

F. Guidi
e-mail: ferruccio.guidi@unibo.it

The important topic of evaluation of systems has been addressed systematically in the literature only recently, and we present the current state in Sect. 7.

The conclusions (Sect. 10) are preceded by some notes on the availability of math retrieval systems (Sect. 8) and some hopefully insightful statistics on the productivity of the research community on the topic, which are presented in Sect. 9.

2 Purpose Driven Taxonomy of Retrieval Problems

The following taxonomy is a categorisation of retrieval problems according to the constraints and expectations of the agent—a human or a program—that issues the query. We will use the taxonomy in the rest of the paper when we will discuss the applicability of the techniques we will propose, keeping in mind that most techniques work best only when applied to one kind of retrieval problem.

When in doubt on the classification of a problem, matching the input, output and constraints should take precedence over matching the general description. For example, the problem of a user that needs to recall some complex formula only remembering an imprecise fragment of it is more likely to be an example of Document Retrieval than of Formula Retrieval, even if it could fit both classifications.

2.1 Problem 1: Document Retrieval

Objective A *human* is interested in recalling a *set of mathematical documents* (or fragments) that are related to a particular mathematical topic. Typically it is not the case that only one document provides the correct answer; on the contrary the user may be interested in a corpus of different documents that yield different, only partially overlapping information. In [31] and other papers there are attempts at a classification of the information needs of users. However, at the moment only the system described in [77] tries to use the classification to improve the user experience.

Input The human composes a query combining keywords (e.g. for topics [1]), free text and mathematical formulae. Often the mathematical formulae are intended as examples of expressions related to the topic of interest. For example, a user interested in trigonometric identities can just enter one identity to retrieve them all. Or a formula showing a particular property of a special function can be used to disambiguate the special function among the ones with similar names.

The query can be composed using a very simple, Google-inspired, single line interface, or written using an ad-hoc query language (see [6,7,72] for some proposals), or by filling in some form. The first solution is the one preferred in the literature. In [31] a comparison of the behaviour of mathematicians vs other users highlighted that the professional mathematician is more interested in the precision of the output than the effort put into the input. Therefore mathematicians may use and appreciate more complex interfaces. On the contrary, other users are likely to prefer a simple, modern search interface.

Formulae can be entered in some textual syntax (e.g. LaTeX, MathML), maybe with the help of on-the-fly formulae rendering [44], or using graphical editors [52], or they can be acquired from hand-written snippets [2]. Some interfaces combine these input methods [74]. The formula is likely to contain errors and ambiguities, for example if it is encoded at the presentational level (e.g. in Presentation MathML or LaTeX), if it is acquired from hand-written text, or if the user only remembers it partially or in a wrong way. Errors and ambiguities are not a critical problem because formulae are just used to retrieve documents that contain *similar* formulae according to some similarity criterion. In [2] the authors address the problem of combining and ranking results from different queries generated from ambiguous formulae due to errors in the recognition process. See also [73] for a survey on the interaction between mathematical information retrieval and mathematical document recognition. Some authors [28] suggest that the visual presentation of the formula may sometimes be important in the definition of similarity, whereas in other situations it is the mathematical *content* of the formula that matters. Logically equivalent formulae whose content encoding is highly different are better considered less similar.

Once the search engine returns the result, the user may be given the opportunity to enhance the query by further filtering.

Output The output is a *ranked list* of matching documents or document fragments (e.g. a chapter of a book, or a section of an article). When the query involves mathematical formulae, the ranking is determined by the similarity relation. The user must be given the possibility to quickly determine whether the matched document is interesting or not. Therefore the problem of how to present summaries of the selected documents in the result list is of fundamental importance [42,44,51,65,67–69,71]. Even highlighting correctly the bits of the summary that matches the query can make a significant difference in the user experience [42,44,67,68]. The list of results must be the starting point for further investigations by the user. At least, all results must contain hyperlinks or other ways to retrieve the original document the summary points to. A study of user requirements in [77] suggests that results should be presented after clustering them according to their resource type (research paper, tutorial, slides, course, book, etc.). For example, a student may immediately decide to skip research papers, and a researcher may skip websites and tutorials.

Constraints A balance must be obtained between *precision* (the fraction of retrieved documents that are relevant) and *recall* (the fraction of relevant documents that are retrieved). To maximise recall, precision is affected and many out of topic documents (false positives) are retrieved, penalising performance. Too many results are overwhelming and the user is likely to give attention only to the first ones in the list. Therefore, the search engine does not have to rank and produce summaries of documents with low scores. The ranking function is ultimately the one responsible for the perceived quality of the search engine.

Since the query is intended to be issued by a human, the performance of the search engine is not a critical requirement and up to a few seconds (or even minutes in some particular situations) may be acceptable. Nevertheless, modern textual search engines like Google are extremely fast, and the user is likely to expect the queries to be solved in less than a second.

2.2 Problem 2: Formula Retrieval

Objective A program—more rarely a human—is interested in retrieving *all* formulae that are in some relation $\mathcal{R}$ with a query formula E. Sometimes the formula E can actually be a set of formulae. For example, E can be a goal to be proved automatically, and $T\ \mathcal{R}\ E$ holds when T is the conclusion of the statement of a theorem that can be instantiated to prove E. More precisely, T contains metavariables to be instantiated and $\mathcal{R}$ is one-sided unification up to some equational theory. The dual query is also used in the literature: E is a property (a statement containing metavariables), $\mathcal{R}$ is unification and the query finds all operations that satisfy the property E. By using several properties at once, the query can find all models of a given theory (e.g. all semirings in the library) [55], also up to renaming of constants and properties. An interesting application presented in [20] that uses techniques similar to [55], consists in matching concepts across libraries by first computing properties of an object in one library (i.e. patterns like commutativity of a binary operator) and then looking for objects in the other library that satisfy the same properties. To be more effective, properties are extracted from all libraries and concepts are matched according to a similarity measure to identify objects that satisfy a similar set of properties. A third example is obtained by choosing logical implication for $\mathcal{R}$. The query looks for all formulae that imply E.

Input One or more formulae E that may or may not contain metavariables to be instantiated. Rarely, additional constraints can be expressed using keywords, classifications, free text, authors, etc. Formulae are not supposed to be ambiguous or contain errors. In [8] ambiguity is resolved before performing the query using type checking and interaction with the user.

Some dedicated query languages are proposed to specify the structure of the formulae E [11–13,23,29,57]. They are implemented on top of relational databases or ad-hoc in-memory indexes.

Output The query is meant to retrieve a set of formulae that satisfy a certain property. When the search is performed by a program, there is no need to present summaries of the document the formula occurs in. Even when a human issued the search, a hyperlink to the document may be sufficient.

In many situations the relation $\mathcal{R}$ can be extended to a ternary relation $T\ \mathcal{R}_\rho\ E$ meaning that T is related to E with score ρ, and the results can be ranked according to ρ. For example, if $\mathcal{R}$ reduces a proof of E to a proof of T, then the Ts may receive a higher score if they are judged easier to prove.

Constraints Maximisation of the recall is fundamental. The query should return all formulae that satisfy the query, even if they rank very lowly. Because searches are often basic operations of complex algorithms (e.g. automatic provers), speed is also critical. In several situations, the searches need to be performed in milliseconds.

To speed up the searches or when the relation $\mathcal{R}$ is undecidable, the search engine may use a second decidable relation $\mathcal{R}'$ such that $\mathcal{R} \subseteq \mathcal{R}'$. Using $\mathcal{R}'$, the query can return false positives, i.e. formulae T such that $T\ \mathcal{R}'\ E$ but not $T\ \mathcal{R}\ E$. For example, when E is a pattern and $\mathcal{R}$ is unification, $\mathcal{R}'$ may ignore the structure of the two formulae E and T and conclude $E\ \mathcal{R}'\ T$ when all symbols in E are also in T. Example: $f(xz, y+z)\ \mathcal{R}\ f(?, ?+?)$ and $f(y+z, xz)\ \mathcal{R}'\ f(?, ?+?)$, but not $f(y+z, xz)\ \mathcal{R}\ f(?, ?+?)$.

2.3 Problem 3: Document Synthesis

Objective Composing a new mathematical document assembling fragments to be retrieved from a library. The most common occurrence is in educational software where a learning object must be assembled according to the expertise of the user, the topic of interest, etc. [14,37,38]. An unrelated example is the automatic generation of summaries and statistics for a mathematical library [12]. A final example is mining of formalised libraries, for example to build visual representations of the graph of dependencies over an axiom (to understand its implications) or a definition/statement (to understand the propagation of changes).

A variant is to solve a mathematical problem by composing mathematical (Web) services [16]. Each service exposes metadata about the problem solved, the algorithm implemented, and its preconditions and postconditions.

Input The query is not likely to involve mathematical formulae, and it is usually expressed using a query language over ontologies. A high level interface may hide the underlying query language. Sometimes the query is fixed once and for all, and needs to be run at regular periods.

Output The expected output depends on the particular use case and it is usually made of a single result in place of a ranked list. The result may consist of a graph of objects and relations between them, or it may consist of the minimal information to build the expected document or solve the algorithmic problem.

Constraints The constraints depend on the particular use case.

2.4 Digression

After an initial screening of the literature, we decided to analyse only papers about the first two problems, where formulae play a central role. Indeed, at a first glance most solutions to Document Synthesis employ standard query languages for ontologies, and only the ontologies themselves are math-specific (e.g. [16,37]). Logic programming languages are also employed to represent what the user knows/ignores and the inference rules to assemble documents [14].

Moreover, we did not find in the literature convincing examples for the need of very expressive query languages to solve the Document Retrieval and the Formula Retrieval problems, where the kinds of queries are essentially fixed a priori. Moreover, evaluation of queries expressed in these languages are reported to be too slow to be used for Formula Retrieval. Sometimes additional techniques are employed for Document Synthesis, like semantic query reduction to relax the user provided query by allowing additional topics close to the one specified by the user [36]. These techniques too seem to be very general and applicable to domains very different from that of mathematics.

Despite the strong interest of the community in the use of formulae in queries, studies on the behaviour of users [32,77] conclude that the added value may be low, and the finding is confirmed in [38] (now 10 years old) and [46] where the logs of the DLMF and of ActiveMath search engines are analysed concluding that only few queries contain mathematical formulae, most are very simple ones, and such queries do not yield satisfactory results. It would be interesting to analyse the logs of the MathWebSearch engine that is now integrated in Zentralblatt Mathematik and the related traffic to see if the negative conclusions are still relevant.

3 Encoding Based Taxonomy

Mathematical information can be encoded in a library at three different levels. The most shallow one is *presentation*. Presentation markup uses a finitary language to express the bi-dimensional layout of a formula, useful to present it to the reader. The standard XML language for presentation is *Presentation MathML*, and several tools can generate Presentation MathML from LaTeX(e.g. LaTeXML,[1] Tralics[2]), PDF files (e.g. Maxtract[3]), handwritten text (e.g. InfTy Reader[4]), digitised documents (e.g. InfTy Reader) or content markup (e.g. via XSLT stylesheets). We can therefore assume that the totality of the documents to be indexed are available in Presentation MathML. Zanibbi and Orakwue [74] contains a small survey on presentation-based math retrieval.

The next level is *content*. At the content level, the structure of the formula is described, and symbols and operators appearing in it are linked to their entry in an ontology, called *content dictionary* in OpenMath[5] terminology. The markup language is finitary, but the ontology is not since new mathematical entities can always be defined. The relation between content and presentation is many-to-many: the same presentation markup may represent different content expressions (*ambiguity*), and a content expression can be given different presentations according to the conventions of the community of readers, the language of the reader, but also for purely aesthetic reasons like constraints on the size of the formula. OpenMath and Content MathML are the two standard XML languages for formulae at the content level. OMDoc[6] is an attempt at standardising at the content level whole mathematical documents, comprising proofs. Content markup is currently mostly used for the exchange of formulae between systems, in particular Computer Algebra Systems. There are no significant examples of large libraries of documents natively written using content markup. Nevertheless, there are tools like SnuggleTeX[7] based on heuristics to semantically enrich (annotated) LaTeX documents or even MathML Presentation documents to content.

The last level is *semantics* and it is specific to libraries of formalised mathematical knowledge. The semantics level refines the content level by picking for every content level object one particular definition in a given logic. The chosen definition embeds the object with additional properties, e.g. computational properties. For example, addition over natural numbers can be defined in the Calculus of Inductive Constructions as a non-computable ternary predicate in logic programming style, or as a recursive function on the first argument—such that $0 + x$ and x become logically indistinguishable—or as a recursive function on the second argument—so that $0 + x$ and x are not indistinguishable, but only provably equal. Interactive theorem provers often provide an XML dump of their internal semantics representation.

Formula Retrieval is expressed either on semantics markup or on content markup. Even when the semantic markup is available, it may be convenient to convert the library to content level by identifying alternative definitions of the same mathematical notion. In this way, it becomes possible to retrieve useful theorems on mathematically equivalent definitions, in the hope to reuse them after conversion to the definitions in use. One application of this technique is reuse of libraries across different systems based on the same logic.

[1] http://dlmf.nist.gov/LaTeXML.

[2] http://www-sop.inria.fr/marelle/tralics/.

[3] https://www.cs.bham.ac.uk/research/groupings/reasoning/sdag/maxtract.php.

[4] http://www.inftyproject.org/en/software.html.

[5] http://www.openmath.org.

[6] http://www.omdoc.org.

[7] http://www2.ph.ed.ac.uk/snuggletex/.

The Document Retrieval problem is expressed independently from the encoding. However, the user is likely to enter formulae in the query using a presentation language (mostly LATEX, even if MathML starts to be used [42,44,51]). Some authors have provided evidence that precision is improved when exploiting parallel markup, even when the content part is automatically generated from the presentation part [53]. See [48] for motivations against content/parallel markup and in favour of a more lightweight encoding of content information in Presentation MathML. Other authors claim that precision can be lost by embracing content because sometimes the actual layout used in a presentation or the name used for variables are significant. González Pinto [21] in retrospect also describes the choice of using Content MathML as a bad decision. Other authors dismiss indexing of Content MathML because of the non-availability of libraries or because of conversion from presentation to content being approximative and unreliable. Finally, [53] reports that automatic conversion of large formulae from Presentation to Content may be computationally unfeasible, and propose to limit the conversion to small ones.

Recently, the debate on presentation only vs parallel markup seems to be solved in favour of the latter. For example, the system that scored better at the NTCIR-11[8] task reports better scores when applied to content markup generated from LATEX as compared to presentation only markup [58]. The authors second this observation already in [43].

Moreover, several works in the literature that deal with presentation markup enrich it—in the document itself or in the indexes—with additional annotations to make explicit additional semantics that is latent in the library [15] or in the text surrounding the formulae [21,34]. For example, in [34] artificial intelligence is applied to the whole document to recover from the text surrounding the formulae the name associated to the mathematical entities in the formula (e.g. "posterior probability", "derivative of f"). González Pinto [21] uses a cheaper approach by considering only one sentence around a formula, but it later observes that one sentence is often not sufficient and many relevant results are therefore missed. Another example is an analysis of co-occurrence of symbols in the corpus to identify related ones. It is shown that these techniques are important to augment recall or, sometimes, precision. In our view, like the heuristic based presentation-to-content translation, these are *attempts to infer and store partial semantics of mathematical expressions*. It may be questioned (see for example [48]) if the current content markups (OpenMath and Content MathML) are the right instruments to augment presentation markup with partial, approximate semantics, and if such additional semantics makes only sense in the indexes of search systems, or it may be serialised to an XML format for being reused by third parties.

Systems based on Content MathML, parallel markup or semantics appear in [11–13,20,24,25,33,41–44,50–52,54,55,57,60,66,75,76]. The remaining ones use Presentation MathML only.

4 Taxonomy of Techniques for Mathematical Retrieval

Implementations of solutions to mathematical search problems can be obtained combining one of the main techniques that will be presented in Sect. 4.2 with a choice of modular enhancement techniques from Sect. 4.1 used to improve precision, recall or both.

4.1 Modular Enhancement Techniques

The following techniques are general enough to be applied to solve both the Document and the Formula Retrieval problems, and by analysing the literature it seems that every system eventually applies all of them.

Segmentation A preliminary step to indexing is segmentation of documents into chunks. Chunks are the unit of information to be returned to the user, with pointers to the parent document. Segmentation is trivial on formal mathematical documents, hard on web-pages, and intermediate on other resources like books or papers [77]. Several

[8] http://research.nii.ac.jp/ntcir/index-en.html.

systems implement segmentation; however, the NTCIR-11 competition has provided a data set of already segmented documents [4] and that may hinder the study of segmentation techniques in the future.

Normalisation To improve recall, both formulae in the query and the formulae in the library are put in normal form before indexing them. Having the same normal form is an equivalence relation $\equiv$, and the query retrieves formulae up to $\equiv$. For Formula Retrieval it is necessary that $\equiv \mathcal{R} \equiv \subseteq \mathcal{R}$. For Document Retrieval the $\equiv$ relation must be compatible with the similarity and ranking functions. If this is not the case, precision can be critically lowered.

Uses of normalisation include: repairing of broken XML/MathML generated by automatic conversion tools [49] (e.g. when the structure imposed by `<mrows>` is not compatible with the mathematical structure); removal of information that does not contribute to the semantics like comments, layout elements (spaces, phantoms and line breaks), XML/MathML attributes (colour, font, elements in other namespaces) [18,27,49]; picking canonical representations of the same presentation/content when different MathML encodings are possible (e.g. `msubsup` vs `msup` and `msub`, `mfenced` vs use of two parentheses, applications of trigonometric functions with/without using parentheses, etc.) [5,18,49]; replacing names of bound variables with unique numerical indexes (e.g. De Bruijn indexes) to search up to α-conversion [49,55]; ignoring parentheses and ordering of arguments of associative/commutative operators [5,50,51,55,62,63,70]; expressing derived notions exposing the derivation (e.g. replacing $x \geq y$ with $y \leq x$, $x \nleq y$ with $\neg(x \leq y)$, arcsin with sin^{-1}, etc.) [5,49]; capturing logical equivalence/type isomorphisms (e.g. writing formulae in prenex normal form, currification of functions) [17,20,55].

A normalised formula can be quite different from the original one, and that can be a symptom that the formula is not significant. Therefore in [58] normalised formulae are weighted according to their similarity to the initial one, and weights are considered during the ranking phase with great results.

Approximation Normalisation does not lose information, converting a document to an equivalent one. Many papers call "normalisation an approximation phase where subformulae are replaced with constrained placeholders to allow the formula to be matched by similarity. For example: names of variables or constants can be replaced by a single name [19]; all numeric constants by a single identifier [19,49,63]; subformulae may be replaced by their type. For example, in [25] type information is used to retrieve formulae by sorted unification, i.e. by constraints with type placeholders in patterns. In [56,74], symbol layout trees are approximated by bags of tuples representing relative positions of symbols. Approximation improves recall. To limit the loss of precision, systems that approximate index both the original and the approximated formula (or even several instances at different levels of approximation). The effects of approximation are similar to those of query reduction, but approximation is more efficient because it works at indexing time.

Enrichment Enrichment works on the library or on the query to augment the information stored/looked for in the index by inferring new knowledge from existing one. It can contribute to the solution of both the Document Retrieval and the Formula Retrieval problems. Typical examples of enrichment are: heuristically generating and storing content metadata from Presentation MathML [48]; automatic/interactive disambiguation of formulae in the queries to perform a precise query at the content or semantics level [8,11–13]; automatic inference of metadata from context analysis or usage analysis (latent semantics) [15,34,65]; automatic extraction of key phrases from mathematical articles [59]. The latter example adapts a general technique to mathematical documents, but de-facto ignoring all formulae.

The most impressive application of enrichment is presented in [26]. The aim is to search for geometrical constructions that are described using a procedural language (e.g. draw the segments connecting A with B, B with C, and A with C). Enrichment consists in replacing the procedural with a declarative description (e.g. ABC is a triangle). The same declarative description can be obtained by multiple procedural ones, and thus recall is greatly improved. The technique can also be seen as a form of normalisation (see Sect. 4.1) where the normal form is not unique (e.g. it may be the case that by analysing the hypothesis one could deduce that ABC is also an equilateral triangle even if that is not stated in the procedural description).

Query reduction Query reduction trades precision for recall by selectively dropping or weakening some of the constraints present in the query. Results obtained from reduced queries can be ranked after results from precise queries. In the literature it occurs in many forms in solutions to both the Formula Retrieval and the Document Retrieval problems: keywords or whole formulae can be omitted from the search (e.g. [45]); a constant can be weakened to other constants that co-occur frequently with the given one; constants that occur too frequently can be dropped from the queries; a formula may be required to match only the toplevel structure of the formula given as a query [9].

An evaluation of different query reduction and result merging strategies on the NTCR-11 dataset is presented in [45].

4.2 Main Techniques

The following techniques are mutually exclusive. Moreover, each technique performs better on only one of the two problems.

Reduction to full-text searches The technology to perform full-text searches is very advanced and there are popular open software implementations with good performance like Apache Lucene[9]/Solr[10] and ElasticSearch.[11] The benefits of reducing search for formulae to full-text searches are speed of execution of the queries and the combination of formula based and textual searches almost for free. The main drawback is that the precise structure of a formula is partially lost in the translations proposed in the literature, and that it is impossible or very hard to capture precisely the kind of relations $\mathcal{R}$ used for Formula Retrieval, unless $\mathcal{R}$ is approximated by a much coarser relation $\mathcal{R}'$. Therefore the technique has been successfully applied so far only to Document Retrieval [1,19,21,27,34,38,42,44,46,47,49–52,56,63,67,69,71,74].

All proposals approximate documents with (large) bags of index terms, whose multiplicity is the term frequency (TF). Index terms can be keywords or "sentences" describing mathematical contents. A bag is represented by the vector of the term frequencies weighted using some variation of inverse document frequency (IDF). This approach assumes that terms appearing in fewer documents of the database are more informative and thus receive higher weight. In particular, the weight u_i of the term i in the document u can be computed with the help of the next formula, where $\mathrm{TF}(u, i)$ is the frequency of the term i in the document u, and $\mathrm{DF}(i)$ is the number of documents containing the term i out of n documents stored in the database.

$$u_i = \mathrm{TF}(u, i) \cdot \log \frac{n}{\mathrm{DF}(i)}$$

The weights u_i are known as the "features" of the document u.

Similarity of two documents u and v is most commonly measured by the cosine of the angle between the vectors u and v. In some cases, the length of these vectors is normalised to speed up the computation of the similarity measure. For example, in [34,64] a sentence is the set (ordered or not) of symbols found in either a path from the root of the formula to a leaf, or as children of the same node. Matching is then performed by a disjunctive query and results are ranked using the just mentioned TF-IDF method with normalised vectors. As the authors claim, the system "is *too* flexible: it is difficult to say where the relevant results stop and random matches begin; thus we predict higher recall but lower precision rates than exact match systems". Other authors extract sentences or n-grams that capture the formula more precisely. As a general remark, the clear impression we got from the literature is that the fewer features extracted, the lower the precision. All kinds of techniques can be used to extract the features, comprising regular expressions [1] and finite state automata [54].

[9] https://lucene.apache.org/.

[10] http://lucene.apache.org/solr/.

[11] https://www.elastic.co/products/elasticsearch.

In order to reduce the number of comparisons made for a query, some systems cluster similar documents hierarchically at indexing time (e.g. [1]), and retrieve documents comparing the feature vector of the query with the centroids of the clusters. For example, documents about trigonometric functions are likely to be automatically clustered together. However most systems do not seem to cluster in advance, and prefer the flexibility of weights to capture similarity of features (e.g. similarity of occurrences of trigonometric functions).

According to the set of features extracted, the weighting function used, and the other modular techniques used in combination, the accuracy achieved by systems based on this technique range from extremely low to extremely high (see, for example [4]).

Structure-Based Indexing via Tries/Substitution Trees Formula Retrieval can be solved with the data structures developed for automatic theorem proving to store libraries of lemmas and quickly retrieve formulae up to instantiation/generalisation. In its simplest form, pointers to all the statements are stored in the leaves of a tree that precisely encodes in its paths the statements. To match a formula, the tree is recursively traversed using the formula to drive the descent. Many variants of this basic approach have been proposed in the Artificial Intelligence literature. They provide different space/time trade-offs, but they are essentially interchangeable in the formula retrieval context.

The relations $\mathcal{R}$ that can be captured are only instantiation and generalisation of whole formulae. These two relations, however, allow to specify more high level queries like searching for all elimination principles for a given data type (the `elim` query of Whelp [8]). MathWebSearch [25,33] and [29] are based on this approach. Retrieval of formulae is very fast, assuming that the index can be entirely stored in main memory.

This approach consistently maximises precision but presents poor recall. To accept larger relations, or to be applied to Document Retrieval, or to cope with too rigid queries, the technique needs to be integrated with other ideas. For example: to match subtrees of formulae in the library, every subtree of a lemma needs to be stored as well in the index; to solve unification problems up to an equational theory that admits normal forms, all formulae are normalised; to allow queries that use keywords or free text, a free-text search engine must be run in parallel and the results need to be combined in the ranking phase [25,37].

Reduction to SQL or Ad-hoc Queries The third approach consists in approximating formulae via relations to be stored in a relational DB [9,23]. An alternative consists in storing the relations in ad-hoc indexes in memory, and it is employed when the indexes already exists for other purposes (typically in libraries of formalised knowledge) [11–13]. The technique is applied to Formula Retrieval and the database can be reused for Document Synthesis without modifications. Approximated queries up to generalisation/instantiation can be made efficient [9] without requiring an index stored in main memory for structure-based indexing. Recall can be maximised by relaxing the representation of formulae as relations or by employing normalisation. Ad-hoc inverted indexes for paths and to map each Content MathML node to its parent have also been used in [24]: the search engine is very fast, but the precision obtained is low.

Reduction to XML-Based Searches Some systems [6,7,72] that index MathML documents at the content level, base their searching capabilities on the existing XPath/XQuery technology. The system described in [60], which is based on Stratosphere, is batch oriented, trades flexibility with performance, and it is essentially math-unaware (for example, it does not normalise the input in any way). Other systems [16,37] that deal with ontologies indexed in the Ontology Web Language, rely on third-party OWL search engines implementing graph matching.

5 Ranking

Because users only inspect the first results returned by a query, precision when solving Document Retrieval is strongly determined by the ranking function. Ranking is also of paramount importance for Formula Retrieval: when the search retrieves the candidates for progressing in a proof, correctly ranking the results may dramatically cut the number of wrong proof attempts and backtracks. The ranking criterion for the two problems is, however, very different: for the first problem similarity of formulae in the query and in the results should contribute significantly to the score; for the second problem the score should be determined by the intended use of the results. For example, a lemma L_1 that exactly matches the goal to prove and has no premise should always score better than a lemma L_2 that also exactly matches the goal, but that has hypotheses to be proved later.

Ranking according to the intended use for Formula Retrieval has received very little interest in the literature we examined. On the other hand, several papers explicitly address ranking for Document Retrieval. The consensus seems to be that a good ranking function needs to be sophisticated and that the usual metrics induced by reduction to textual searches are completely inadequate (see, for example [69]). All the proposed ranking techniques are strongly based on heuristics and, unfortunately, most of them are incomparable and hard to combine.

One class of metrics takes into consideration also the structure of the formulae involved and the enriched semantics, when available. For example [75] heavily exploits Content MathML to rank results by considering the taxonomic distance of constants (where close is approximated to being defined in the same content dictionary), the data type hierarchical level (matching a function is more significant than matching a numerical constant), matching depth (partially matching the formula at the top level is more significant than matching a deeply nested subformula), coverage (percentage of formula matched), kind of matched expression (formula vs term). All this information needs to be computed and amalgamated employing a heuristic algorithm that depends on a set of parameters optimised in [76]. A second paper [61] confirmed that each one of the listed similarity feature factors significantly improves the ranking, but the last one that still contributes, has lower relevance.

In [63] ranking is determined by the weights used during matching, and the authors claim that each document base and scientific field should have its own weighting function. Nevertheless, they “tried to create a complex and robust weighting function that would be appropriate to many fields”.

In [69] the author proposes a parametrised ranking function that works on mathematical documents (not only formulae) that seems applicable to enriched presentation and that weights a lot of additional information including keywords, the number of cross-references and their kind (e.g. definitional vs propositional). Ranking employs a hybrid of scalarisation and vectorisation.

In [30] the authors propose to adopt tree edit distance to measure similarity of formulae. Most of the paper is about optimisations to improve efficiency of ranking because tree edit distance is hard to compute. The final proposal combines some clever memorisation and a procedure to quickly prune documents bounding their similarity scores with a lightweight computation. The paper also shows benchmarks comparing the processing time and success rate of most search and ranking algorithms in the literature, reimplemented by the authors and run on the same dataset. From the benchmarks the method proposed seems to be superior, but the implementations do not exploit relevant enriched information like cross-references, semantic proximity of definitions, etc. The benchmarks are therefore non conclusive.

In [54] the authors employ a continuous learning ranking model after having extracted features from Content MathML mathematical formulae using a finite state automata. Benchmarks show their ranking to be superior than the ones used in classical ranking of textual documents. However, they do not compare with [30] or [69] (that work on Presentation MathML).

Simpler approaches to ranking can be found in [66] (based on Subpath Set, reported to work well only on “simplified” Content MathML) and [28] that works on Presentation MathML and measures similarity as a function of the size of subtrees in common. The ranking metric used in [1] is a TF-IDF modified with weights to assign more importance to some operators, but the details given to determine the weights are insufficient.

Ranking algorithms can be too complex to be incorporated in the search phase, for example when using Lucene technology. Moreover, they are typically slower than the search phase. Therefore several authors suggest to re-rank only the first results of the query that employs a simpler ranking measure to determine the interesting candidates to be ranked more accurately [30,69].

An algorithm to automatically categorise documents is presented in [77], where it is argued that clustering documents according to their category greatly improves the usefulness of the tool for the user.

6 Architecture of a Mathematical Document Retrieval System

Figures 1 and 2 show a reference architecture for the document retrieval system that adopts all the modular techniques of Sect. 4.1 and reduction to full text search as the main technique (see Sect. 4.2). The architecture is essentially shared by all the systems that are built around the Lucene/Solr or ElasticSearch.

Transformations are grouped into boxes when they can be re-ordered or applied zero or multiple times. Translations between presentation languages in Fig. 1 are subsumed by normalisation; the possible translations from presentation to content or parallel markup is subsumed by semantic enrichment. Disambiguation is the counterpart of semantic enrichment for queries. Therefore the architecture can accommodate all kind of encodings.

The indexing phase ends with the creation of the feature vectors that are stored in the full-text-search database. The indexer and retrieval engines can just be off-the shelf, customised popular implementations like Lucene. The remaining components can be stand-alone, but some of them, like the feature extractors or the re-rankers, are better implemented as customisations of the Lucene-like software.

It seems to us premature to propose a reference architecture for formula retrieval. Interactive theorem provers that directly integrate formula retrieval heavily exploit their existent, sometimes ad-hoc data structures. It is unclear if they could benefit from a generic implementation. The general purpose systems that have been well developed

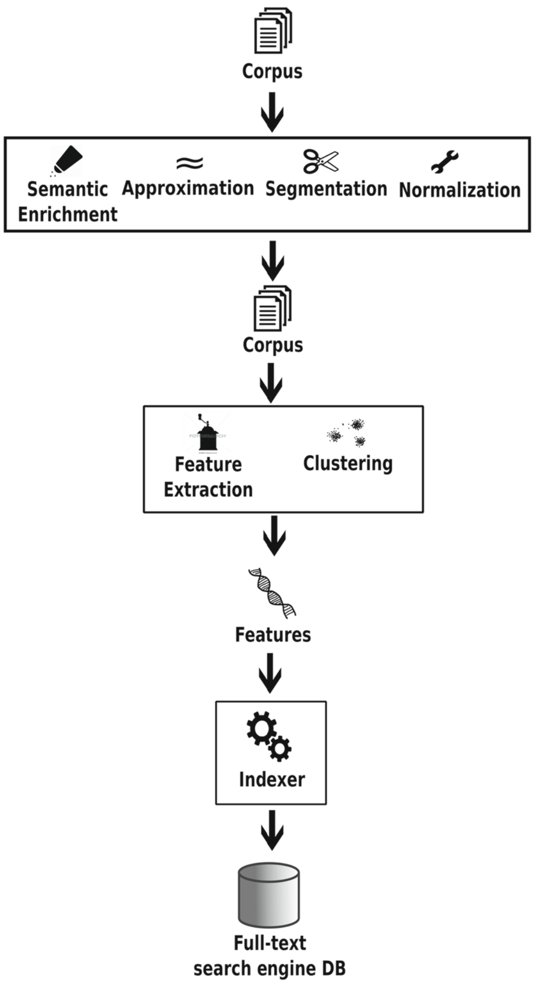

Fig. 1 Document retrieval via reduction to full text searches: indexing

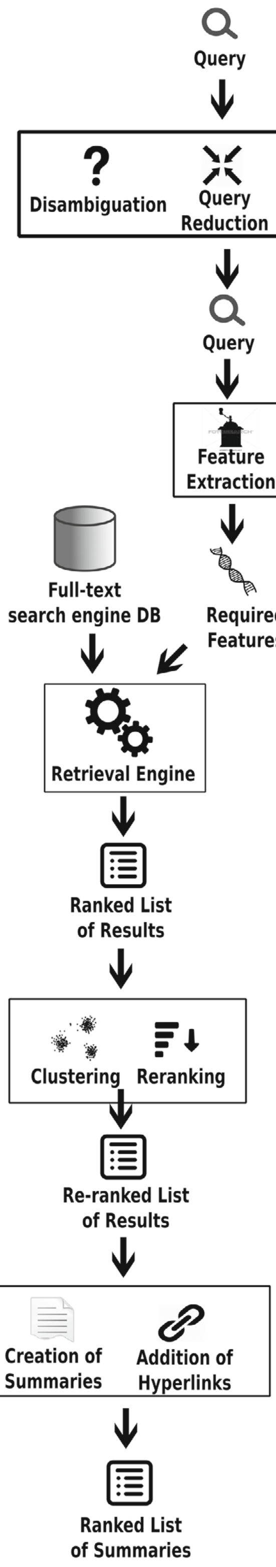

Fig. 2 Document retrieval via reduction to full text searches: retrieval

do not share the same main technique: MathWebSearch [33] employs structure-based indexing, Whelp [9] reduces the queries to SQL. Both approaches benefit from the modular techniques we identified, but the techniques are integrated at different times. For example, approximation is an optional pre-processing step for structure-based indexing, and it integrates the computation of all sub-formulae. It is instead an intrinsic step of feature extraction in the work of Asperti et al. [9,23]. In the latter case, no computation of sub-formulae is required. However, the second approach requires a mandatory refinement step on the approximated results of the query, while the refinement step may be skipped in structure-based indexing.

7 Evaluation of Math Information Retrieval

Several papers present benchmarks on the systems proposed, and rarely (e.g. [30]) compare them with reimplementations of the algorithms found in the literature. The significance of most of these benchmarks is unclear, because conflicting results are found in the literature, most techniques are not presented in sufficient details in the papers to be exactly reproduced, and systems are very sensitive to the kind of queries examined. The only alternative is to compare different tools on unbiased, standard benchmarks that are currently lacking.

The main issue is not to come up with large corpora of documents: at least for Document Retrieval on enriched Presentation MathML documents, a large corpus can be easily obtained converting documents from ArXiV, DLMF, PlanetMath, Wikipedia, etc. For Formula Retrieval, the existing libraries of interactive theorem provers, like Mizar and Coq, can be directly used after conversion. The problem is to determine large sets of *real world, interesting queries*, and to evaluate the results. Automatic evaluation is particularly hard in the domain of mathematics, whereas manual evaluation is limited to a tiny number of queries and runs. Formulating good sets of queries is also complex, because users with different mathematical background and motivations are likely to issue different queries. Moreover, what makes a query hard can just be the use of non standard mathematical notations, errors in the encoding of formulae, or formulation at the wrong level of abstraction. Líška [41] discusses the problem at length and reviews the state of the art of evaluation of Math Information Retrieval before 2013, including the experience of the MIR workshop at CICM 2012 were two systems were compared on about ten hard queries proposed by the judges, and the conclusion was that the systems were too sensitive to the formulation of the query.

The situation is improving since 2013 with the creation of a math oriented task in the NTCIR initiative [3,4] that is attracting a small, but increasing number of participants [19,21,24,25,33–35,39,43,56,58,60,61,64]. The initiative is too young to come to definite conclusions and the current choice of tasks and queries is not granted yet to have significant coverage and to be unbiased. For example, in [34] the authors report that despite several improvements to the tool (quantified via NTCIR-11 runs), their tool scored lower than in NTCIR-10. They justify the phenomenon by noticing that "in NTCIR-11, query variables get much bigger emphasis, most topics feature complete and very particular formulae, and sub-formulae matching is not nearly as useful as before". Indeed, as reported in [4] "the design decision …to exclusively concentrate on formula/keyword queries and use paragraphs as retrieval units …has also focused research away from questions like result presentation and user interaction. …few of the systems have invested into further semantics extraction from the data set. …We feel that this direction should be addressed more in future challenges". An effect of the bias towards search up to unification w.r.t. search up to similarity is observable in [56,74] too: the system proposed works very well even if it works on Presentation MathML only and the set of features extracted is very simple (bag-of-symbol-pairs model, where a pair is made of two symbols in a father-son relation). The reason why it works well is that the authors also index approximated pairs where the child is a wildcard, and in future works they are thinking at improving even more the handling of wildcards to score better. In comparison, most other systems based on feature vectors just replace wildcards in the query with `<m:ci>` identifiers. The emphasis on formula queries is also to be evaluated considering the already cited works that conclude that users do not see (yet?) much value in them [32,77].

Finally, the NTCIR task does not cover distinctly the Document Retrieval and the Formula Retrieval problems, but only Document Retrieval with an emphasis on exact pattern matching of formulae that should be more distinctive of Formula Retrieval.

8 Availability of Math Retrieval Systems

Most of the systems described in the literature are research prototypes, and the majority of them are no longer working or no longer accessible. At the time this paper was written, the only ones for Document Retrieval with a running Web interface or code that can be downloaded are: (1) Design Science's MathDex[12] (formerly MathFind) (2) NIST DMLF[13] (3) MathWebSearch,[14] which is also used by the Zentralblatt Mathematik[15] repository (4) MiAS[16] (Math Indexer and Searcher), also used to search the EuDML[17] (5) Tangent[18] In addition to those, the following commercial systems are also accessible: (a) Springer LaTeXSearch[19] (b) Wolfram Alpha[20] Systems (a) and (2) index LaTeX directly; (1) and (5) are based on Presentation MathML; systems (3) and (4) can use either Presentation MathML or Content MathML/parallel markup, but work better on the latter; finally system (b) actually generates on the fly most of the result of the query, for example by plotting functions, computing their Taylor expansion, etc. It does not really qualify then as a search engine.

The two most comprehensive reviewing services for mathematics, namely Zentralblatt Mathematik and MathSciNet,[21] integrate a search engine. While the former has integrated MathWebSearch to mix formulae, keywords and text in the queries, the latter has no support for formulae.

Most interactive theorem provers also have their own implementation of a search engine to solve Formula Retrieval. Most of the time, the implementation is embedded in the system and does not work on the whole library at once, with the exception of MML Query[22] for Mizar.

9 Productivity of the Research Community

The still unsatisfactory performance of math retrieval systems in terms of accuracy and precision can be puzzling if compared with the large number of publications. The same for the low number of available research systems. In this Section we try to shed some light on the phenomenon running some statistics on the papers.

Our first analysis aims at determining if the problem is still under investigation, and also if research on mathematical retrieval is sustained or it occurs in bursts. Figure 3 plots the cumulative number of publications per year. The curve growths almost linearly up to 2008, decreases its slope from 2008 to 2012, and finally becomes steep again after 2012. The latest speed up is certainly due to the NTCIR competition that has attracted new interest and new research groups. The intermediate slow-down is also quite understandable, even if worrisome. On the one hand it is due to a relatively sufficient level of maturity reached for formula retrieval queries issued by humans: most developers of interactive theorem provers addressed the problem and stopped working on it when they found a solution that was performing well enough. On the other hand, many people working on document retrieval have initially explored techniques that did not perform well, and abandoned the field. Significant recent progress after 2008 is mostly due to the steady work of a very few groups that keep improving their own software.

To collect evidence for our last claim we have performed a second analysis, whose result is shown in Fig. 4. We have taken the papers in the literature and we have clustered them according to co-authorship: we initialise each

[12] http://www.mathdex.org.

[13] http://dlmf.nist.gov.

[14] http://search.mathweb.org.

[15] https://zbmath.org/.

[16] https://mir.fi.muni.cz/mias.

[17] http://eudml.org.

[18] https://www.cs.rit.edu/~dprl/Software.html#tangent.

[19] http://latexsearch.com.

[20] https://www.wolframalpha.com/.

[21] http://www.ams.org/mathscinet.

[22] http://mmlquery.mizar.org.

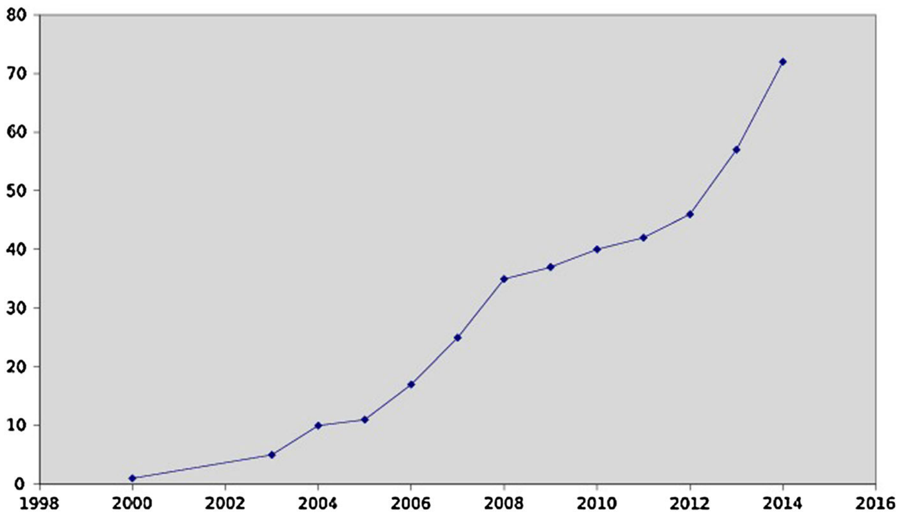

Fig. 3 Cumulative number of publications per year

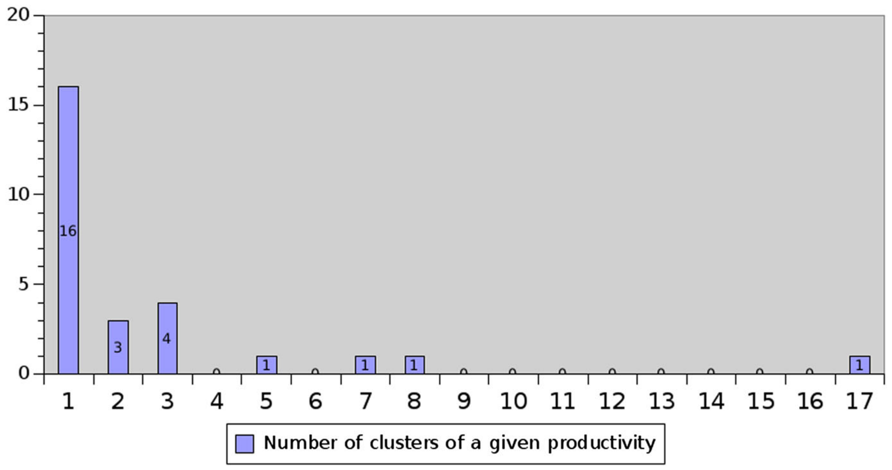

Fig. 4 Analysis of the productivity of research groups

cluster with only one paper and we keep merging two clusters when there are two papers—one per cluster—that have at least one author in common. Before performing the clustering we manually excluded the papers that describe the NTCIR task that are clearly co-authored by people that belong to research groups that did not collaborate otherwise so far. To each cluster we assign its productivity, i.e. the number of papers collectively published. Finally, we plot the number of clusters that reached a certain productivity.

The Figure shows that 16 papers have had no follow up by the authors that seem to have lost interest in the problem after writing a single paper. Similarly, only 3 clusters have co-authored 2 papers and 4 clusters 3 papers, showing a moderate continuous interest in the topic. At the other side of the spectrum there is the research group of Abdou Youssef that has authored 17 papers, mostly on the topic of document retrieval. The second larger cluster is the research group of Michael Kohlhase that has authored 8 papers, moving its focus from formula retrieval to document retrieval. It is closely followed by the group of Martin Líška with 7 papers, again mostly on document retrieval, and by the one of the authors with 5 papers on formula retrieval only.

Table 1 quantifies the number of authors of papers in the field. It shows that the majority of papers have two authors only, despite the topic being an applied one. Only 10 papers have more than 3 authors. Combined with the

Table 1 Number of authors per paper

Number of authors	Number of papers
1	13
2	34
3	13
4	6
5	3
6	1

Table 2 Average number of authors per paper and average size of a cluster as a function of the productivity of the cluster

Productivity of the cluster	Number of clusters	Average number of members	Average number of authors per paper
17	1	10.0	1.9
8	1	7.0	2.0
7	1	4.0	2.4
5	1	7.0	2.6
3	4	4.8	1.6
2	3	3.0	2.8
1	16	2.4	2.4

previous analysis, this one suggests that developing a math retrieval system does not require a lot of manpower at once, but a sustained effort to progressively refine the solution.

To further sustain our claim, we refine the analysis in Table 2 by focusing on the productivity of research groups. The table shows both the average size of clusters that have a given productivity and the average number of authors per paper published by the cluster. More productive clusters are likely to have more members, but the average numbers of authors per paper seems independent from the productivity. Note that the first two columns of Table 2 are the data plotted in Fig. 4.

According to the previous analyses, the NTCIR and similar initiatives that promote continuous refinement of a system are the most likely to have a positive impact on the field. However, we also expect some of the groups that participated in the competition and that achieved bad results to retire in future ones. Regarding projects funding, local funding that can sustain long term refinements are the most likely to have a positive impact. The outcome of large collaborative projects, on the other hand, is more dubious: while different groups could contribute expertise in the modular techniques of Sect. 4.1, the average number of authors per paper suggests that there is no issue of manpower, and that we are still in a phase where competition is to be preferred to standardisation.

10 Conclusions

Mathematical knowledge retrieval, the low hanging fruit of Mathematical Knowledge Management, is still far from being grasped. Despite the significant amount of work dedicated to the topic in the last 12 years, only a few systems are still available, and their precision and recall scores compared to other knowledge retrieval fields are low. Moreover, usability and user requirement studies suggest that queries containing formulae—the main focus of the majority of papers—are perceived by users as not very useful (yet?).

The main contributions of this paper have been providing a hopefully comprehensive bibliography on the subject, and presenting taxonomies for both mathematical retrieval problems and techniques. We believe that our purpose driven taxonomy can be useful in classifying papers, in clarifying the scope of application of techniques and in the much needed development of unbiased benchmarks for mathematical retrieval.

References

1. Adeel, M., Cheung, H.S., Khiyal, S.H.: Math GO! prototype of a content based mathematical formula search engine. J. Theor. Appl. Inf. Technol. **4**(10), 1002–1012 (2008)
2. Ahmadi, S.A., Youssef, A.: Lexical error compensation in handwritten-based mathematical information retrieval. In: DML 2008. Towards digital mathematics library, Birmingham, UK, July 27th, 2008. Proceedings, pp. 43–54. Masaryk University, Brno (2008)
3. Aizawa, A., Kohlhase, M., Ounis, I.: NTCIR-10 math pilot task overview. In: Proceedings of the 10th NTCIR conference, Tokyo, pp. 654–661 (2013)
4. Aizawa, A., Kohlhase, M., Ounis, I.: NTCIR-11 math 2 task overview. In: Proc. 10th NTCIR conference (Tokyo, Japan), pp. 88–98 (2014)
5. Altamimi, M.E., Youssef, A.: A more canonical form of content MathML to facilitate math search. In: Proc extreme markup languages (2007)
6. Altamimi, M.E., Youssef, A.S.: Wildcards in math search, implementation issues. In: Proceedings of the ISCA 20th international conference on computer applications in industry and engineering, CAINE 2007, November 7–9, 2007, San Francisco, pp. 90–96 (2007)
7. Altamimi, M.E., Youssef, A.S.: A math query language with an expanded set of wildcards. Math. Comput. Sci. **2**(2), 305–331 (2008)
8. Asperti, A., Guidi, F., Coen, C.S., Tassi, E., Zacchiroli, S.: A content based mathematical search engine: Whelp. In: Types for proofs and programs, pp. 17–32. Springer (2006)
9. Asperti, A., Selmi, M.: Efficient retrieval of mathematical statements. In: Mathematical knowledge management, third international conference, MKM 2004, Bialowieza, September 19–21, 2004, Proceedings, pp. 17–31 (2004)
10. Asperti, A., Zacchiroli, S.: Searching mathematics on the web: state of the art and future developments. In: FIZ, editor, Proceedings of New Developments in Electronic Publishing of Mathematics, pp. 9–18 (2004)
11. Bancerek, G.: Information retrieval and rendering with MML Query. In: Mathematical knowledge management, pp. 266–279. Springer (2006)
12. Bancerek, G., Rudnicki, P.: Information retrieval in MML. In: Mathematical knowledge management, second international conference, MKM 2003, Bertinoro, February 16–18, 2003, Proceedings, pp. 119–132 (2003)
13. Bancerek, G., Urban, J.: Integrated semantic browsing of the mizar mathematical library for authoring mizar articles. In: Mathematical knowledge management, third international conference, MKM 2004, Bialowieza, September 19–21, 2004, Proceedings, pp. 44–57 (2004)
14. Baumgartner, P., Furbach, U.: Automated deduction techniques for the management of personalized documents. Annals of Mathematics and Artificial Intelligence **38**(1–3), 211–228 (2003)
15. Cairns, P.A.: Informalising formal mathematics: searching the mizar library with latent semantics. In: Mathematical knowledge management, third international conference, MKM 2004, Bialowieza, September 19–21, 2004, Proceedings, pp. 58–72 (2004)
16. Caprotti, O., Dewar, M., Turi, D.: Mathematical service matching using description Logic and OWL. In: Mathematical knowledge management, 3rd international conference, MKM 2004, Bialowieza, September 19–21, 2004, Proceedings, pp. 73–87 (2004)
17. Delahaye, D.: Information retrieval in a Coq proof library using type isomorphisms. In: Types for proofs and programs, pp. 131–147. Springer (2000)
18. Formánek, D., Líška, M., Růžička, M., Sojka, P.: Normalization of digital mathematics library content. In: Davenport, J., Jeuring, J., Lange, C., Libbrecht, P. (eds.) Joint Proceedings of the 24th OpenMath Workshop, the 7th workshop on mathematical user interfaces (MathUI), and the work in progress section of the conference on intelligent computer mathematics, number 921 in CEUR Workshop Proceedings, pp. 91–103, Aachen (2012)
19. Gao, L., Wang, Y., Hao, L., Tang, Z.: ICST math retrieval system for NTCIR-11 Math-2 Task. In: Proc. 10th NTCIR conference (Tokyo, Japan), pp. 99–102 (2014)
20. Gauthier, T., Kaliszyk, C.: Matching concepts across HOL Libraries. In: Intelligent computer mathematics—international conference, CICM 2014, Coimbra, July 7–11, 2014. Proceedings, pp. 267–281 (2014)
21. González Pinto, J.M., Barthel, S., Balke, W.-T.: QUALIBETA at the NTCIR-11 math 2 task: an attempt to query math collections. In: Proc. 10th NTCIR conference (Tokyo, Japan), pp. 103–107 (2014)
22. Guidi, F., Coen, C.S.: A survey on retrieval of mathematical knowledge. In: Kerber, M., Carette, J., Kaliszyk, C., Rabe, F., Sorge, V., (eds.) Intelligent computer mathematics—international conference, CICM 2015, Washington, DC, USA, July 13–17, 2015, Proceedings, Lecture Notes in Computer Science, vol. 9150, pp. 296–315. Springer (2015)
23. Guidi, F., Schena, I.: A query language for a metadata framework about mathematical resources. In: Mathematical knowledge management, second international conference, MKM 2003, Bertinoro, February 16–18, 2003, Proceedings, pp. 105–118 (2003)

24. Hagino, H., Saito, H.: Partial-match retrieval with structure-reflected indices at the NTCIR-10 math task. In: Proc. 10th NTCIR conference (Tokyo, Japan), pp. 692–695 (2013)
25. Hambasan, R., Kohlhase, M., Prodescu, C.: MathWebSearch at NTCIR-11. In: Proc. 10th NTCIR Conference (Tokyo, Japan), pp. 114–119 (2014)
26. Haralambous, Y., Quaresma, P.: Querying geometric figures using a controlled language, ontological graphs and dependency lattices. In: Intelligent computer mathematics—international conference, CICM 2014, Coimbra, Portugal, July 7–11, 2014. Proceedings, pp. 298–311 (2014)
27. Hashimoto, H., Hijikata, Y., Nishida, S.: Incorporating breadth first search for indexing MathML objects. In: Systems, Man and Cybernetics, 2008. SMC 2008. IEEE international conference on, pp. 3519–3523 (2008)
28. Kamali, S., Tompa, F.W.: Improving mathematics retrieval. In: DML 2009. Towards digital mathematics library, Grand Bend, Ontario, Canada, July 8–9th 2009. Proceedings, pp. 37–48. Masaryk University, Brno (2009)
29. Kamali, S., Tompa, F.W.: A new mathematics retrieval system. In: Proceedings of the 19th ACM international conference on information and knowledge management. CIKM '10, pp. 1413–1416. ACM, New York (2010)
30. Kamali, S., Tompa, F.W.: Structural similarity search for mathematics retrieval. In: Intelligent computer mathematics—MKM, calculemus, DML, and systems and projects 2013, Held as Part of CICM 2013, Bath, July 8–12, 2013. Proceedings, pp. 246–262 (2013)
31. Kohlhase, A.: Search interfaces for mathematicians. In: Intelligent computer mathematics—international conference, CICM 2014, Coimbra, July 7–11, 2014. Proceedings, pp. 153–168 (2014)
32. Kohlhase, A., Kohlhase, M.: Re examining the MKM value proposition: from math web search to math web re search. In: Towards mechanized mathematical assistants, 14th Symposium, Calculemus 2007, 6th International Conference, MKM 2007, Hagenberg, June 27–30, 2007, Proceedings, pp. 313–326 (2007)
33. Kohlhase, M., Prodescu, C.: MathWebSearch at NTCIR-10. In: Proc. 10th NTCIR conference (Tokyo, Japan), pp. 675–679 (2013)
34. Kristianto, G.Y., Topić, G., Ho, F., Aizawa, A.: The MCAT math retrieval system for NTCIR-11 math track. In: Proc. 11th NTCIR conference (Tokyo, Japan), pp. 120–126 (2014)
35. Larson, R.R., Reynolds, C.J., Gey, F.C.: The abject failure of keyword IR for mathematics search: Berkeley at NTCIR-10 Math. In: Proc. 10th NTCIR conference (Tokyo, Japan), pp. 662–666 (2013)
36. Libbrecht, P.: Escaping the trap of too precise topic queries. In: Intelligent computer mathematics—MKM, calculemus, DML, and systems and projects 2013, Held as Part of CICM 2013, Bath, July 8–12, 2013. Proceedings, pp. 296–309 (2013)
37. Libbrecht, P., Desmoulins, C., Mercat, C., Laborde, C., Dietrich, M., Hendriks, M.: Cross-curriculum search for intergeo. In: Intelligent computer mathematics, 9th international conference, AISC 2008, 15th Symposium, Calculemus 2008, 7th international conference, MKM 2008, Birmingham, July 28–August 1, 2008. Proceedings, pp. 520–535 (2008)
38. Libbrecht, P., Melis, E.: Methods to access and retrieve mathematical content in activemath. In: Mathematical software-ICMS 2006, pp. 331–342. Springer (2006)
39. Lipani, A., Andersson, L., Piroi, F., Lupu, M., Hanbury, A.: TUW-IMP at the NTCIR-11 Math-2. In: Proc. 11th NTCIR conference (Tokyo, Japan), pp. 143–146 (2014)
40. Líška, M.: Searching Mathematical Texts. Bachelor's thesis, Masaryk University, Faculty of Informatics, Brno (2010)
41. Líška, M.: Evaluation of Mathematics Retrieval. Master's thesis, Masaryk University, Faculty of Informatics, Brno (2013)
42. Líška, M., Sojka, P., Líška, M., Mravec, P.: Web interface and collection for mathematical retrieval: WebMIaS and MREC. In: DML 2011. Towards digital mathematics library, Bertinoro, July 20–21st, 2011. Proceedings, pp. 77–84. Masaryk University, Brno (2011)
43. Líška, M., Sojka, P., Růžička, M.: Similarity search for mathematics: Masaryk University team at the NTCIR-10 Math Task. In: Proc. 10th NTCIR Conference (Tokyo, Japan), pp. 686–691 (2013)
44. Líška, M., Sojka, P., Růžička, M.: Math indexer and searcher web interface—towards fulfillment of mathematicians' information needs. In: Intelligent computer mathematics—international conference, CICM 2014, Coimbra, Portugal, July 7–11, 2014. Proceedings, pp. 444–448 (2014)
45. Líška, M., Sojka, P.: Růžička, M.: Combining text and formula queries in math information retrieval: evaluation of query results merging strategies. In: Proceedings of the first international workshop on novel web search interfaces and systems. NWSearch '15, pp. 7–9. ACM, New York (2015)
46. Miller, B.R.: Three years of DLMF: Web, Math and Search. In: Intelligent computer mathematics—MKM, Calculemus, DML, and systems and projects 2013, Held as Part of CICM 2013, Bath, July 8–12, 2013. Proceedings, pp. 288–295 (2013)
47. Miller, B.R., Youssef, A.: Technical Aspects of the digital library of mathematical functions. Ann. Math. Artif. Intell. **38**(1–3), 121–136 (2003)
48. Miller, B.R., Youssef, A.: Augmenting presentation MathML for Search. In: Intelligent computer mathematics, 9th international conference, AISC 2008, 15th symposium, calculemus 2008, 7th international conference, MKM 2008, Birmingham, July 28–August 1, 2008. Proceedings, pp. 536–542 (2008)
49. Miner, R., Munavalli, R.: An approach to mathematical search through query formulation and data normalization. In: Towards mechanized mathematical assistants, 14th symposium, calculemus 2007, 6th international conference, MKM 2007, Hagenberg, June 27–30, 2007, Proceedings, pp. 342–355 (2007)
50. Misutka, J., Galambos, L.: Mathematical extension of full text search engine indexer. In: Information and communication technologies: from theory to applications, 2008. ICTTA 2008. 3rd international conference on, pp. 1–6 (2008)

51. Mišutka, J., Galamboš, L.: Extending full text search engine for mathematical content. In: DML 2008. Towards digital mathematics library, Birmingham, July 27th, 2008. Proceedings, pp. 55–67. Masaryk University, Brno (2008)
52. Munavalli, R., Miner, R.: Mathfind: a math-aware search engine. In: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval, pp. 735–735. ACM (2006)
53. Nghiem, M.-Q., Kristianto, G.Y., Topić, G., Aizawa, A.: Which one is better: presentation-based or content-based math search? In: Intelligent computer mathematics—international conference, CICM 2014, Coimbra, July 7–11, 2014. Proceedings, pp. 200–212 (2014)
54. Nguyen, T.T., Chang, K., Hui, S.C.: A math-aware search engine for math question answering system. In: Proceedings of the 21st ACM international conference on information and knowledge management, pp. 724–733. ACM (2012)
55. Normann, I., Kohlhase, M.: Extended formula normalization for *epsilon*-retrieval and sharing of mathematical knowledge. In: Towards mechanized mathematical assistants, 14th symposium, Calculemus 2007, 6th international conference, MKM 2007, Hagenberg, June 27–30, 2007, Proceedings, pp. 356–370 (2007)
56. Pattaniyil, N., Zanibbi, R.: Combining TF-IDF text retrieval with an inverted index over symbol pairs in math expressions: the tangent math search engine at NTCIR 2014. In: Proc. 11th NTCIR conference (Tokyo, Japan), pp. 135–142 (2014)
57. Rabe, F.: A query language for formal mathematical libraries. In: Intelligent computer mathematics—11th international conference, AISC 2012, 19th symposium, Calculemus 2012, 5th international workshop, DML 2012, 11th international conference, MKM 2012, systems and projects, Held as Part of CICM 2012, Bremen, July 8–13, 2012. Proceedings, pp. 143–158 (2012)
58. Růžička, M., Sojka, P., Líška, M.: Math indexer and searcher under the hood: history and development of a winning strategy. In: Proc. 11th NTCIR conference (Tokyo, Japan), pp. 127–134 (2014)
59. Schöneberg, U., Sperber, W.: POS tagging and its applications for mathematics—text analysis in mathematics. In: Intelligent computer mathematics—international conference, CICM 2014, Coimbral, July 7–11, 2014. Proceedings, pp. 213–223 (2014)
60. Schubotz, M., Leich, M., Markl, V.: Querying large collections of mathematical publications: NTCIR10 math task. In: Proc. 10th NTCIR Conference (Tokyo, Japan), pp. 667–674 (2013)
61. Schubotz, M., Youssef, A., Markl, V., Cohl, H.S., Li, J.J.: Evaluation of similarity-measure factors for formulae based on the NTCIR-11 Math Task. In: Proc. 10th NTCIR conference (Tokyo, Japan), pp. 108–113 (2014)
62. Shatnawi, M., Youssef, A.: Equivalence detection using parse-tree normalization for math search. In: Second IEEE international conference on digital information management (ICDIM). December 11–13, 2007, pp. 643–648. Lyon, Proceedings (2007)
63. Sojka, P., Líška, M.: The art of mathematics retrieval. In: Proceedings of the 11th ACM symposium on document engineering, pp. 57–60. ACM (2011)
64. Topić, G., Kristianto, G.Y., Nghiem, M.-Q., Aizawa, A.: The MCAT math retrieval system for NTCIR-10 math track. In: Proc. 10th NTCIR conference (Tokyo, Japan), pp. 680–685 (2013)
65. Wolska, M., Grigore, M.: Symbol declarations in mathematical writing. In: DML 2010. Towards digital mathematics library, Paris, July 7–8th, 2010. Proceedings, pp. 119–127. Masaryk University, Brno (2010)
66. Yokoi, K., Aizawa, A.: An approach to similarity search for mathematical expressions using MathML. In: DML 2009. Towards digital mathematics library, Grand Bend, Ontario, Canada, July 8–9th 2009. Proceedings, pp. 27–35. Brno: Masaryk University (2009)
67. Youssef, A.: Search of mathematical contents: issues and methods. In: Proceedings of the ISCA 14th international conference on intelligent and adaptive systems and software engineering. July 20–22, 2005, pp. 100–105. Novotel Toronto Centre, Toronto (2005)
68. Youssef, A.: Roles of math search in mathematics. In: Mathematical knowledge management, 5th international conference, MKM 2006, Wokingham, August 11–12, 2006, Proceedings, pp. 2–16 (2006)
69. Youssef, A.: Methods of relevance ranking and hit-content generation in math search. In: Towards mechanized mathematical assistants, 14th Symposium, Calculemus 2007, 6th international conference, MKM 2007, Hagenberg, June 27–30, 2007, Proceedings, pp. 393–406 (2007)
70. Youssef, A., Shatnawi, M.: Math search with equivalence detection using parse-tree normalization. In: The 4th international conference on computer science and information technology (2006)
71. Youssef, A.S.: Relevance ranking and hit description in math search. Math. Comput. Sci. **2**(2), 333–353 (2008)
72. Youssef, A.S., Altamimi, M.E.: An extensive math query language. In: 16th international conference on software engineering and data engineering (SEDE-2007). July 9–11, 2007, pp. 57–63. Imperial Palace Hotel Las Vegas, Las Vegas, Proceedings (2007)
73. Zanibbi, R., Blostein, D.: Recognition and retrieval of mathematical expressions. Int. J. Doc. Anal. Recognit. (IJDAR) **15**(4), 331–357 (2012)
74. Zanibbi, R., Orakwue, A.: Math search for the masses: multimodal search interfaces and appearance-based retrieval. In: Kerber, M., Carette, J., Kaliszyk, C., Rabe, F., Sorge, V., (eds.), Intelligent computer mathematics—international conference, CICM 2015, Washington, DC, July 13–17, 2015, Proceedings, Lecture notes in computer science, vol. 9150, pp. 18–36. Springer (2015)
75. Zhang, Q., Youssef, A.: An approach to math-similarity search. In: Intelligent computer mathematics—international conference, CICM 2014, Coimbra, July 7–11, 2014. Proceedings, pp. 404–418 (2014)
76. Zhang, Q., Youssef, A.: Performance evaluation and optimization of math-similarity search. In: Kerber, M., Carette, J., Kaliszyk, C., Rabe, F., Sorge, V., (eds.) Intelligent computer mathematics—international conference, CICM 2015, Washington, DC, July 13–17, 2015, Proceedings, Lecture notes in computer science, vol. 9150, pp. 243–257. Springer (2015)
77. Zhao, J., Kan, M.-Y., Theng, Y.L.: Math information retrieval: user requirements and prototype implementation. In: Proceedings of the 8th ACM/IEEE-CS joint conference on digital libraries, pp. 187–196. ACM (2008)